LEZIONE 24

DLX SEQUENZIALE

Nel DLX l'esecuzione di tutte le istruzioni può essere scomposta in 5 passi, ciascuno eseguito in uno o più cicli di clock.

Tali passi sono detti:

- FETCH – L'istruzione viene prelevata dalla memoria e posta in IR
- DECODE – L'istruzione in IR viene decodificata e vengono prelevati gli operandi sorgente dal Register File
- EXECUTE – Elaborazione aritmetica o logica mediante la ALU
- MEMORY – Accesso alla memoria e, nel caso di BRANCH, aggiornamento del PC (non tutte le istruzioni ne hanno bisogno). Le istruzioni che non accedono alla memoria in questa fase non fanno nulla.
- WRITE-BACK – Scrittura sul Register File.
  In generale la maggior parte delle istruzioni indicano che venga aggiornato un registro destinazione.

La divisione in queste cinque fasi ci permette di ottenere una certa efficienza.

FETCH

Il program counter viene inviato al MAR e viene letto dalla memoria il contenuto del MAR. Se c'è la connessione diretta tra il PC e le memorie non si passa dal MAR ma si legge direttamente all'indirizzo presente nel PC.

MAR<-PC; IR<-M[MAR]

DECODE

Oltre a far partire la decodifica, il DLX estrae preventivamente dal Register File (che ha due porte di lettura) due registri (due ipotetici Rs1 e Rs2), li mette in due buffer (registri A e B) e mentre fa questa cosa si porta direttamente avanti incrementando di 4 il program counter.

A <- RS1, B <- RS2, PC <- PC+4

Le fasi successive dipendono molto dall'istruzione da eseguire

EXECUTE

La decodifica è finita e quindi il processore sa di che tipo di istruzione si tratta.

ISTRUZIONE DI ACCESSO ALLA MEMORIA

E' necessario calcolare l'indirizzo (è ottenuto in un solo modo, cioè la somma del contenuto di un registro sorgente e un immediato a 16 bit esteso con segno a 32 bit).

Se è necessario scrivere, è necessario prendere il secondo registro sorgente, che viene messo nel Memory Data Register (MDR)

OPERAZIONI ALU

Viene eseguita l'operazione ALU prevista dall'istruzione stessa. Gli immediati possono essere due registri sorgente o un registro sorgente e un immediato. La ALU viene programmata dall'unità di controllo per eseguire l'operazione prevista. Dopo aver preparato gli operandi, la ALU compie l'operazione.

ISTRUZIONI BRANCH

Se il BRANCH non è preso, il prossimo fetch verrà fatto a PC+4, altrimenti il fetch verrà fatto all'indirizzo che corrisponde a PC+4 sommato all'immediato a 16 bit esteso a 32 bit. Il fatto che il BRANCH sia taken o no viene analizzato dall'unità di controllo.

ISTRUNZIONI JUMP E JUMP AND LINK

L'unica differenza con le istruzioni di BRANCH non bisogna fare nessuna verifica sul fatto che il salto sia preso o no. Nel caso di J/JAL con immediato, l'immediato a 26 bit viene esteso a 32 bit, se invece la J/JAL viene fatta con registro, si salta direttamente all'indirizzo indicato nel registro stesso.

MEMORY

ACCESSO ALLA MEMORIA

LOAD – Viene letto cosa c'è nel MAR e viene caricato in MDR

STORE – Viene letto cosa c'è in MDR e viene caricato nel MAR

BRANCH

Al termine della fase di memory il registro sorgente sa se il registro era zero o no e sa se dovrà saltare all'indirizzo calcolato nella fase di execute o se dovrà proseguire a PC+4

JAL e JALR

Utilizzano la fase di memory perché bisogna salvare in R31 il Program Counter (già incrementato a PC+4), che viene messo nel registro temporaneo C, che poi ovviamente sarà messo in R31

WRITE-BACK

ISTRUZIONI J, JR, JAL, JALR

Viene messo nel PC l'indirizzo a cui è necessario fare il salto e, se è una JAL/JALR, viene scritto in R31 il contenuto di C (cioè il PC già incrementato di 4)

ALTRE ISTRUZIONI

Viene aggiornato il registro destinazione, se l'istruzione prevede che ciò sia fatto.

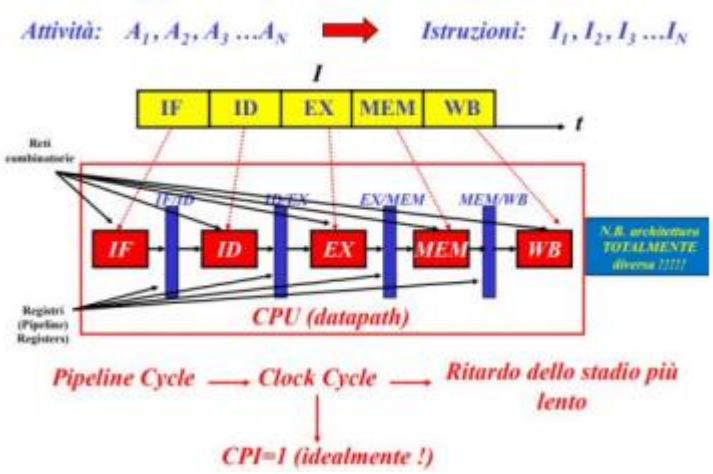
DLX PIPELINED

E' un modo semplice per realizzare un processore più efficiente con molti più vantaggi. E' la norma in tutti i processori.

Il principio base del pipelining è la catena di montaggio.

Il tempo necessario per attraversare la catena di montaggio è detto LATENCY (latenza).

La frequenza con cui vengono completate le attività è detta THROUGHPUT.



Le barriere blu sono necessarie per mantenere le informazioni strettamente necessarie per eseguire l'istruzione stessa. Le barriere blu non sono altro che dei registri edge-triggered. Le barriere separano le fasi combinatorie.

Scegliamo come tempo minimo il caso peggiore richiesto dai cinque stadi per poter uniformare il tempo di esecuzione dei cinque stadi.

La prima istruzione richiederebbe cinque clock, ma dopo questi, nel caso ideale, ogni istruzione viene terminata a ogni clock.

Per calcolare il tempo di un ciclo di clock dobbiamo considerare il tempo di setup dei registri edge triggered.

$$T_{CLK} = T_D + T_P + T_{SU}$$

T_{CLK} – Clock cycle

T_D – Ritardo registro a monte

T_P – Ritardo stadio combinatorio più lento

T_{SU} – Setup registro a valle

REQUISITI PER L'IMPLEMENTAZIONE IN PIPELINE

- Ogni stadio deve essere attivato in ogni ciclo di clock
- E' necessario incrementare il PC in IF invece che in ID
- E' necessario introdurre un ADDER (PC <- PC + 4 – PC <- PC + 1) nello stadio IF
- Sono necessari due MDR (LMDR e SMDR) per gestire il caso di una LOAD seguita immediatamente da una STORE (WB – MEM sovrapposti, sovrapposizione di due dati in attesa di essere scritti, uno in memoria e l'altro nel Register File)

- In ogni ciclo di clock devono poter essere eseguiti 2 accessi alla memoria (IF, MEM): Istruction Memory (IM) e Data Memory (DM) (ARCHITETTURA HARVARD)
- IM e DM devono essere delle memorie cache (on-chip)
- I Pipeline Registers trasportano sia dati sia informazioni di controllo (l'unità di controllo è distribuita fra gli stadi della pipeline e non è più monolitica)

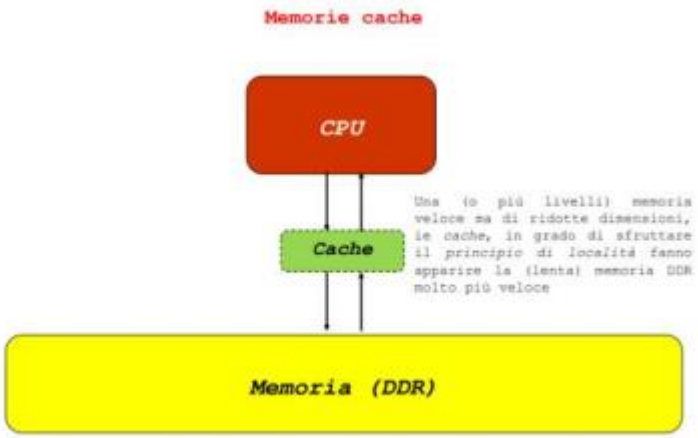
Il principio di funzionamento delle memorie cache è il principio di località. Vengono tenute vicine le cose che potrebbero servire.

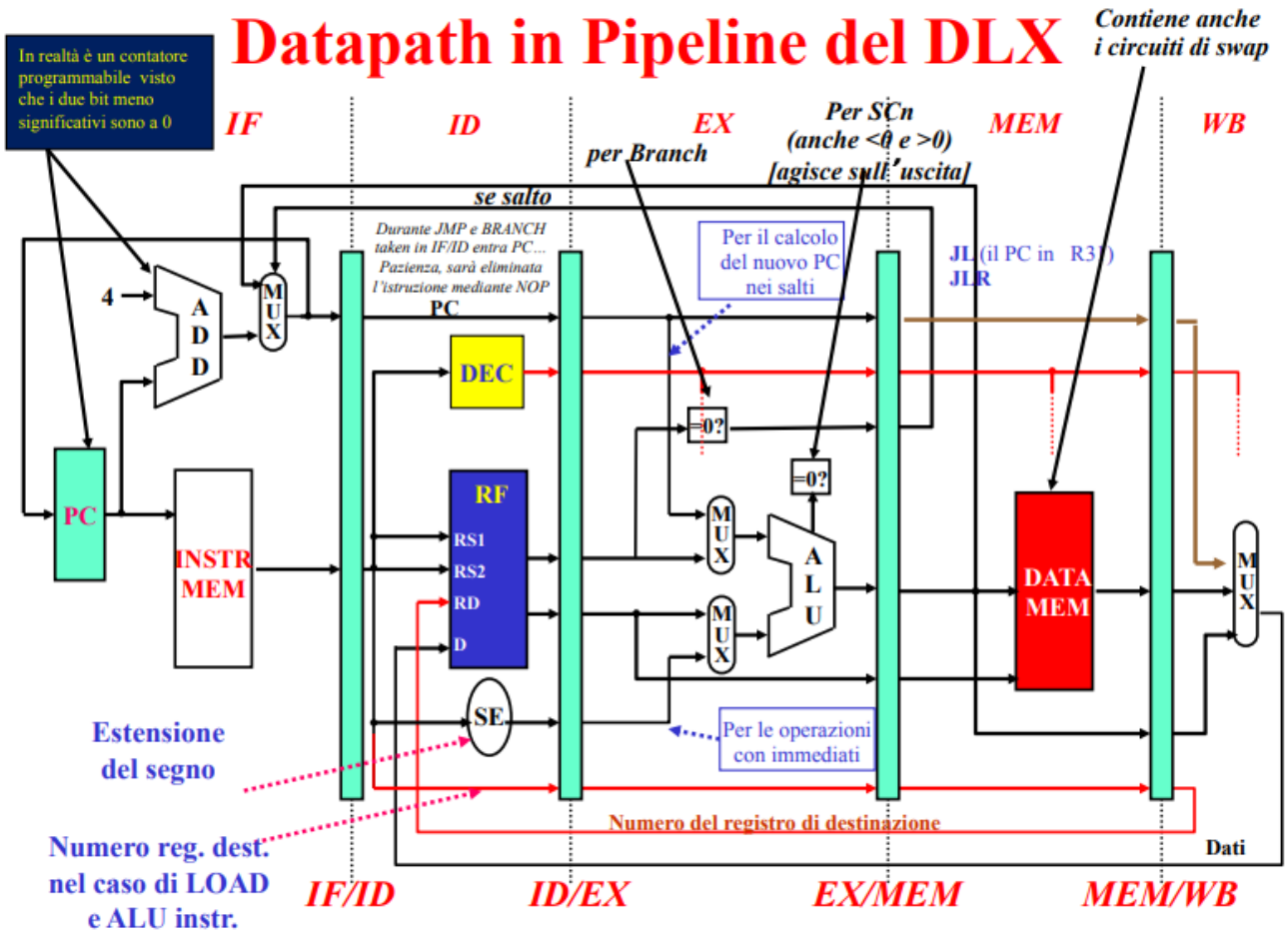
La cache è rappresentata da L1, L2, L3. Aumentando la dimensione della cache aumenta anche il tempo di accesso.

Le cache sono fatte con dei LATCH. Un bit su una cache costa molto di più di un bit su una RAM, ma può essere prelevato in tempi molto più rapidi.

Nelle cache si cercano gli elementi per il loro indirizzo. Se il microprocessore vuole accedere a un certo dato, le varie cache vengono interrogate sul fatto che abbiano o meno quel dato, a cui è associato quel determinato indirizzo. E' il controllore di memoria a interrogare le cache.

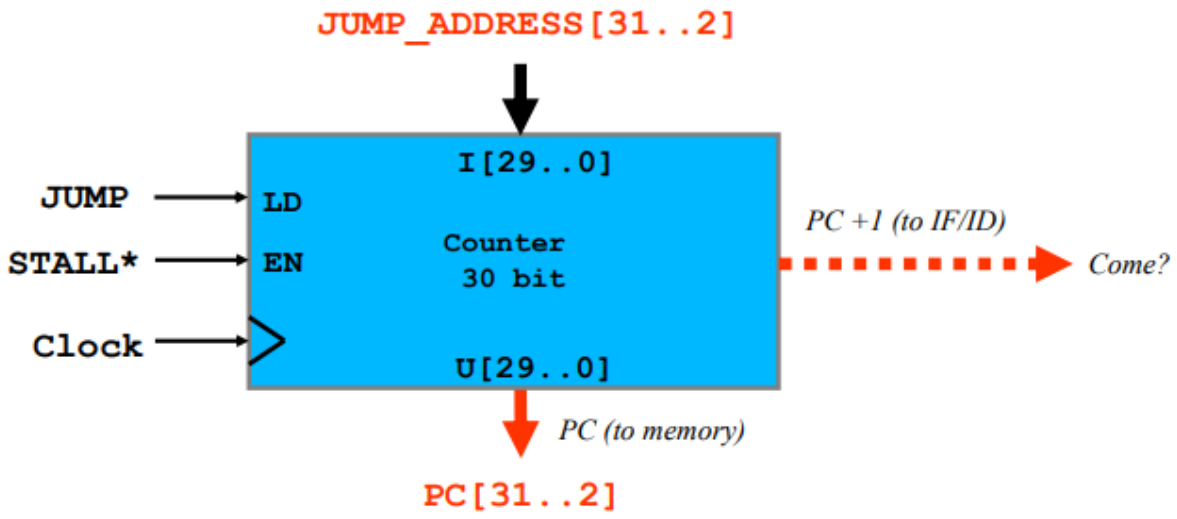
Se la cache non ha il dato si va a controllare la memoria che sta più in basso nella gerarchia. Quando trova l'elemento, il controllore va a riempire i livelli più alti della gerarchia con elementi vicini al dato che è stato richiesto.





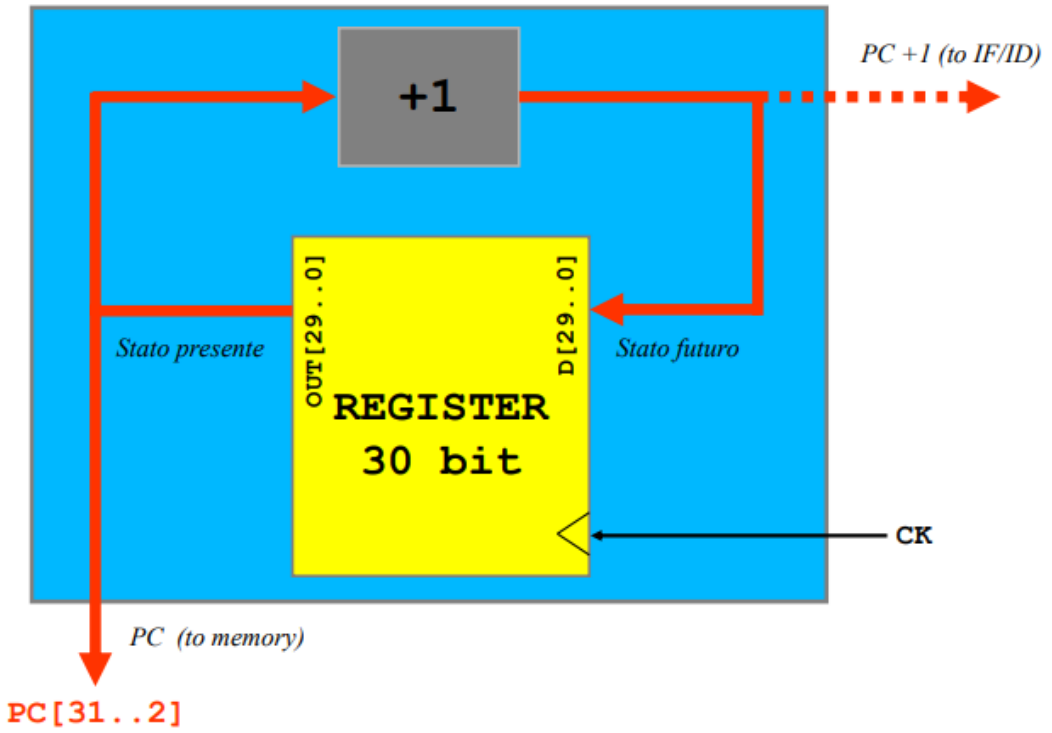
L'aggiornamento del PC avviene con un clock di ritardo rispetto all'uscita; perciò, nella fase di decode e nella fase di execute arrivano delle istruzioni che non dovrebbero esserci, e che quindi andrebbero eliminate. Per renderle innocue si cambia il loro codice operativo prima che modifichino qualcosa.

Possiamo rimpiazzare la fase di fetch con un contatore a 30 bit dato che PC+4 diventa PC+1.

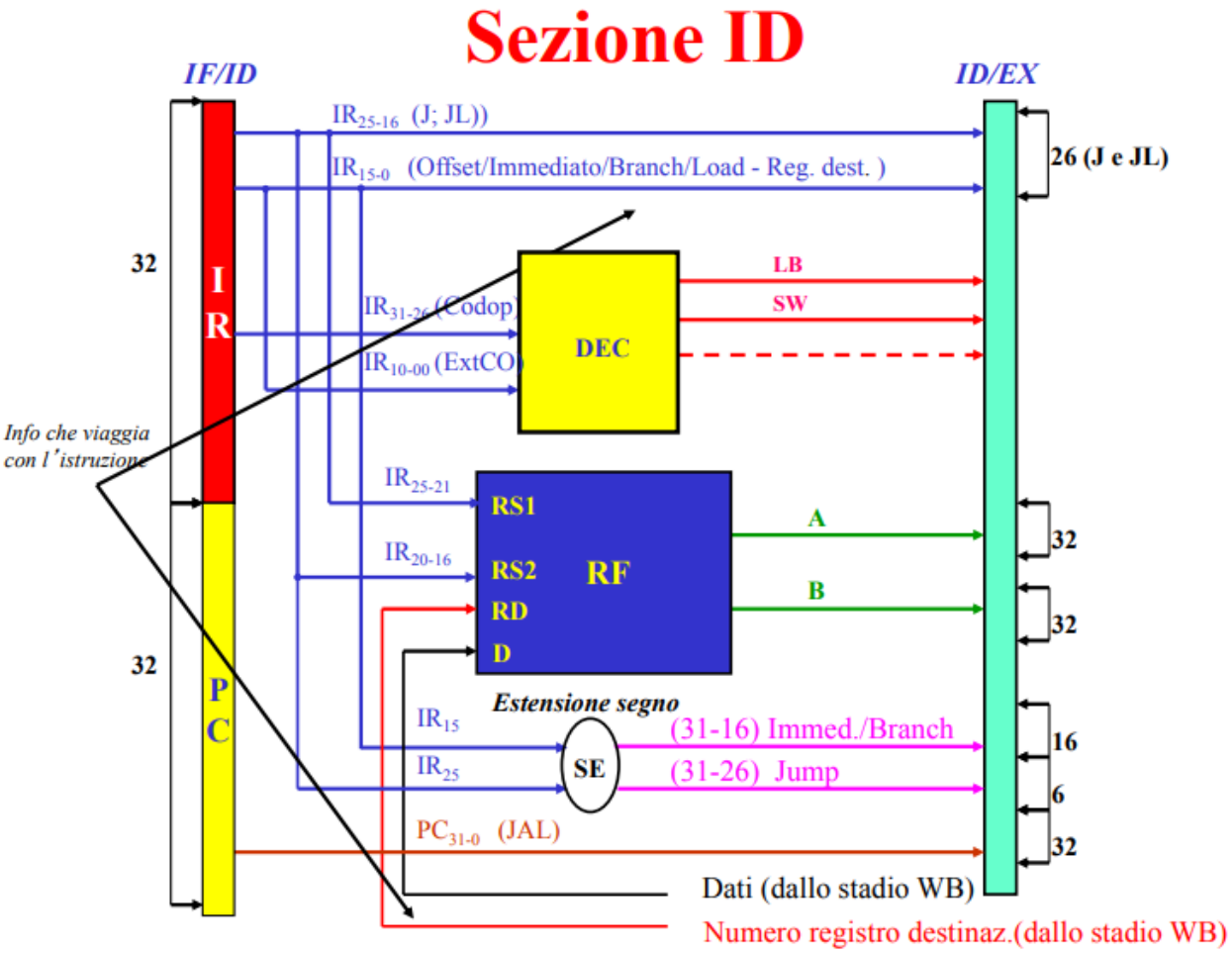


Il Load è prioritario rispetto all'enable, quindi se è 1 viene caricato l'indirizzo di destinazione del salto. Il counter incrementa a meno che non si abbia necessità di fermare la pipeline.

Un'osservazione: come possiamo generare PC + 1 per lo stadio IF/ID (quando viene eseguito il fetch a PC è necessario fare entrare nella pipeline (stadio IF/ID) PC +1 ?



FASE ID



Contemporaneamente alla decodifica vengono estratti i due registri A e B. Il DLX va anche a scrivere nel registro destinazione, attraverso l'informazione che arriva dallo stadio di Write-back.

ALEE NELLE PIPELINE

Si verifica una situazione di alea quando in un determinato ciclo di clock un'istruzione presente in uno stadio della pipeline non può essere eseguita in quel clock.

Alee strutturali – Una risorsa è condivisa fra due stadi della pipeline: le istruzioni che si trovano correntemente in tali stadi non possono essere eseguite contemporaneamente. (Es. se non avessimo due ALU negli stadi di fetch ed execute avremmo un problema perché nello stesso istante due stadi dovrebbero usare la ALU)

Alee di dato – Sono dovute a dipendenze fra le istruzioni. Ad esempio, una istruzione che legge un registro scritto da un'istruzione precedente (RAW)

Alee di controllo – Le istruzioni che seguono un branch dipendono dal risultato del branch (taken o not taken).

L'istruzione che non può essere eseguita viene bloccata, insieme a tutte quelle che la seguono, mentre le istruzioni che la precedono avanzano normalmente (così a rimuovere la causa dell'alea).

FORWARDING UNIT

Se il dato è in WB e l’istruzione è in ID, RF leggerebbe il dato non corretto, perciò si interviene con un MUX